Exp No. 6

Handwritten Digit Recognition using Back Propagation

Srinidhi. V
18CL02

Aim:

To detect the handwritten digit images using ANN with p' back propagation algorithm.

Algorithm:

* Loading the MNIST dataset (grey scale preprocessed images) into three lists two containing training data and third containing test data using pickle Load().
* Filling the 28*28 input image into a numpy array of 784*1 dimension.
* Converting the single output into numpy array of 10 dimensions with 1 at the index of the output an 0.
* Randomly initialising weights mapping input layer & hidden layer & hidden layer to output layer.
* Training the data using back propagation algorithm.
* Forward propagate by calculating the output with randomly assigned weight values using activation function - Sigmoid function.
* Calculate the error in the output.
* Back propagate and gradient descent to find how much hidden layer contributed in our missing output.

* Updating the values of weights by how much missed.
* Return the new weights.
* Trained to ready to test the test dataset
* Test data to tested with above trained network; weights and accuracy of model is calculated.

Sample Input And Output:

MNIST DATASET of handwritten digit images of around 60,000.

Accuracy of network model trained using ANN with backpropagation : 92.5%.

Program:

Digit_reg.py:

```
import numpy as np import mnist_loader as ml import Network as net import testi as tt
import bar as br np.random.seed(1)
weights = 2*np.random.random((784,50)) - 1 weights1
= 2*np.random.random((50,10)) - 1
tr_data, val_data, test_data = ml.load_data()
tr_inputs = [np.reshape(x, (784, 1)) for x in tr_data[0]]
tr_outputs = [ml.vectorized_result(x) for x in tr_data[1]]
for i in range(50000):
    weights , weights1 = net.train(tr_inputs[i],tr_outputs[i],weights,weights1)
    if(i % 500) == 0 :
        br.progress(i, 50000)
br.progress(50000, 50000, cond = True)
print ("\n") print ("Network Trained and ready to be

operated")

te_inputs = [np.reshape(x, (784,1)) for x in test_data[0]]
te_outputs = test_data[1]
tt.check(te_inputs,te_outputs,weights,weights1)
```

mnist_loader.py:

```
import pickle import
gzip import numpy
as np
def load_data(): f = gzip.open('mnist.pkl.gz', 'rb') training_data,

    validation_data, test_data = pickle.load(f, encoding = 'latin1') f.close()

    return (training_data, validation_data, test_data)

def vectorized_result(j):
    e = np.zeros((10, 1))
    e[j] = 1.0 return e
```

Network.py:

```
import numpy as np
def sigmoid(x): return
   1/(1+np.exp(-x))
def deriv_sigmoid(x): return x*(1-x) def

train(inputs,output,weights,weights1):

   x = inputs.T y
   = output.T
   l1 = sigmoid(np.dot(x,weights)) l2
   = sigmoid(np.dot(l1,weights1))
   error = y - l2
   l2_del = error * deriv_sigmoid(l2)
   error0 = l2_del.dot(weights1.T)
   l1_del = error0 * deriv_sigmoid(l1)
   weights1 += np.dot(l1.T,l2_del)

   weights += np.dot(x.T,l1_del) return

   weights,weights1
```

bar.py:

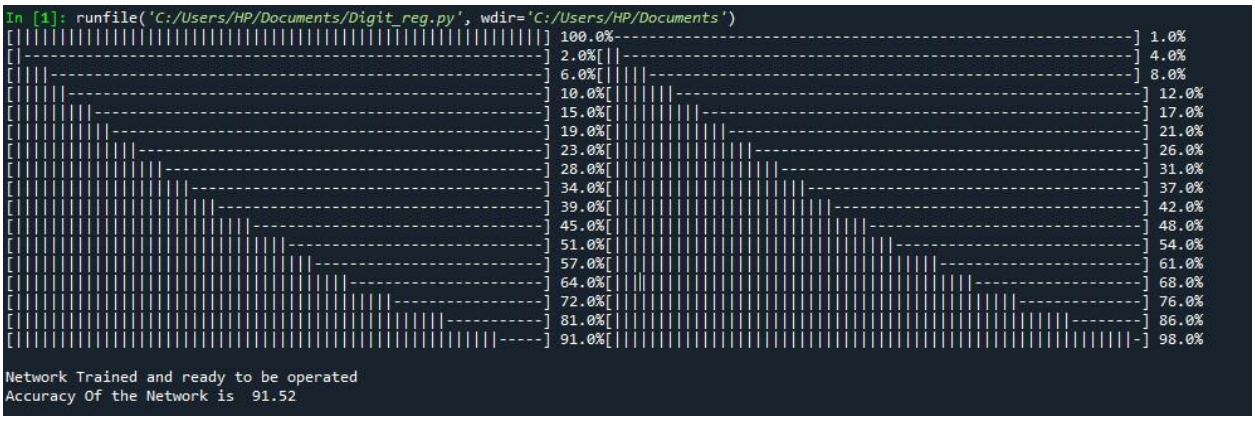
```
import sys
def progress(count, total, cond=False):
   bar_len = 60 filled_len = int(round(bar_len * count
   / float(total)))
   percents = round(100.0 * count / float(total), 1)
   bar = '|' * filled_len + '-' * (bar_len - filled_len)
   if cond == False: sys.stdout.write('[%s] %s%s\r' % (bar,
         percents, '%')) sys.stdout.flush()
   else: sys.stdout.write('[%s] %s%s' % (bar, percents, '%'))
```

testi.py:

```
import Network as net
import numpy as np
def feedforward(x,weights,weights1):
   l = x.T l1 = net.sigmoid(np.dot(l,weights)) l2 =

net.sigmoid(np.dot(l1,weights1)) return l2; def

check(te_inputs,te_outputs,weights,weights1):

   correct = 0
```

```
for i in range(len(te_inputs)):
    out = feedforward(te_inputs[i],weights,weights1) f_out =
np.argmax(out) if(f_out == te_outputs[i]): correct += 1 print
("Accuracy Of the Network is " , ((correct/10000)*100))
```

Output:



Result:

Thus using MNIST dataset handwritten digit recognition is implemented using ANN with backpropagation algorithm.